Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.
A Modelling Language for Rich Internet Applications

A thesis presented in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

at Massey University, Turitea,
New Zealand.

Jevon Michael Wright

2011
Abstract

This thesis presents the Internet Application Modelling Language (IAML), a modelling language to support the model-driven development of Rich Internet Applications (RIAs). This definition includes a visual syntax to support the graphical development of IAML model instances, and the underlying metamodel satisfies the metamodeling and viewpoint architectures of the Model Driven Architecture.

While there are many existing modelling languages for web applications, none of these languages were found to be expressive enough to describe fundamental RIA concepts such as client-side events and user interaction. This thesis therefore presents IAML as a new language that reuses existing standards where appropriate. IAML is supported by a proof-of-concept CASE tool within the Eclipse framework, and released under an open source license to encourage industry use. This reference implementation successfully integrates a number of different model-driven technologies to demonstrate the expressiveness of the modelling language.

The IAML metamodel supports many features not found in other web application modelling languages, such as Event-Condition-Action rules; the expression of reusable patterns through Wires; and a metamodel core based on first-order logic. Through the implementation of the RIA benchmarking application Ticket 2.0, the concepts behind the design of IAML have been shown to simplify the development of real-world RIAs when compared to conventional web application frameworks.
Acknowledgments

This thesis would not have been possible without the support of many important people.

In particular, to my supervisors Jens Dietrich and Giovanni Moretti – thank you for all of your advice, encouragement and inspiration throughout my undergrad degree and my doctoral studies. I am gracious for the patience and support that you have both given me.

To my office mates at Massey that have helped me throughout these years – Graham Jenson, Fahim Abbasi, Yuliya Bozhko and others – thank you for your feedback, ideas and your constructive criticisms; and the innumerable breaks for caffeine, whether in coffee, chocolate or cola form.

To the support staff at Massey – including Patrick Rynhart, Dilantha Pushithewa, and Michele Wagner – thank you for helping me out with the stress and effort necessary to deal with the administrative side of research.

To all my bandmates – Aaron Badman, Aaron Shirriffs, Ian Luxmoore, Josh Williamson, Matt Carkeek, Steve Starr, Wayne Bryant – thank you for supporting me throughout my busy schedules, and for giving me a noisy outlet to express my creativity far away from the thesis.

To all of my family, especially my mum and dad – thank you for your never-ending and reliable support and patience throughout all of my decisions, and encouraging my independence.

And finally to my best friend and fiancé, Krystle Chester, you have helped me and supported me through more than I could have imagined, and completing this thesis would not have been possible without your love, patience and mews. <3
# Contents

Abstract i
Acknowledgments iii
Contents v
List of Tables vii
List of Figures ix
List of Listings xi

## 1 Introduction

1.1 Web Applications .................................... 1
1.2 Modelling ........................................ 3
1.3 Research Questions ................................... 3
1.4 Research Design .................................... 4
1.5 Research Method .................................... 6
1.6 Thesis Outline ...................................... 7

## 2 Background

2.1 Rich Internet Applications ............................... 9
  2.1.1 Classifications of Rich Internet Applications ................. 11
  2.1.2 Interactivity ................................... 11
  2.1.3 Enterprise Software ............................... 12
  2.1.4 Security Risks of Rich Internet Applications .................. 12
2.2 Frameworks ....................................... 13
2.3 Features of Rich Internet Applications ......................... 14
  2.3.1 Feature Categories ................................ 14
  2.3.2 Detailed Modelling Requirements ....................... 17
2.4 Existing Web Application Modelling Languages .................... 20
  2.4.1 WebML ..................................... 20
  2.4.2 UML ...................................... 21
  2.4.3 UWE ...................................... 22
  2.4.4 WebDSL .................................... 23
  2.4.5 Web Information Systems ........................... 24
  2.4.6 Older Languages ................................ 24
## Contents

2.4.7 Evaluation ................................... 24
2.5 Benchmarking Application ............................... 24
2.6 Metrics ......................................... 26
   2.6.1 Web Application Metrics ............................ 26
   2.6.2 Metamodelling Metrics ............................. 28
   2.6.3 System Metrics ................................ 29
2.7 Software Engineering Principles within the Development of Modelling Languages ..... 30
   2.7.1 Modelling Language Design Approaches ................... 30
   2.7.2 Software Process Models ............................ 31
   2.7.3 Test Driven Development ............................ 33
   2.7.4 Open Source .................................. 33
2.8 Conclusion ....................................... 33

3 Modelling ....................................... 35
   3.1 Modelling ........................................ 35
      3.1.1 Metamodels ................................... 36
      3.1.2 Domain-Specific Languages .......................... 36
      3.1.3 Model-Driven Development .......................... 38
      3.1.4 Model-Driven Engineering ........................... 38
      3.1.5 Model Driven Architecture .......................... 39
      3.1.6 Model Space .................................... 43
      3.1.7 Model Transformations ............................. 44
      3.1.8 Query/Views/Transformations Language ................. 46
      3.1.9 The Syntax and Semantics of Metamodels ................... 48
      3.1.10 A Formal Definition of Metamodels ..................... 49
      3.1.11 The Bigger Picture of Modelling Approaches ................. 50
      3.1.12 Metamodelling Environments .......................... 51
   3.2 Model Completion ................................... 51
      3.2.1 Documented Conventions ........................... 52
      3.2.2 Model Completion Concepts .......................... 52
      3.2.3 Model Completion Semantics .......................... 53
   3.3 Model Instance Verification ............................... 54
      3.3.1 Validation vs. Verification ........................... 55
      3.3.2 A Formal Definitions of Constraints ...................... 55
      3.3.3 Discussion ................................... 57
   3.4 Visual Modelling .................................... 58
      3.4.1 Visual Metaphors ................................ 58
      3.4.2 Visual Metaphors in Existing Models ...................... 59
      3.4.3 Visual Modelling Software .......................... 59
      3.4.4 Visual Notation .................................. 59
   3.5 Conclusion ....................................... 61
4 Rich Internet Application Modelling Concepts

4.1 Managing Complexity through Structural Decomposition
   4.1.1 Modularity
   4.1.2 Hierarchical Modelling
   4.1.3 Aspects
   4.1.4 Discussion

4.2 Events
   4.2.1 UML
   4.2.2 AORML
   4.2.3 URML
   4.2.4 Kaitiaki
   4.2.5 Event Algebra
   4.2.6 DOM Events

4.3 Operation Modelling
   4.3.1 UML Activity Diagrams
   4.3.2 Business Process Modelling Notation
   4.3.3 Predicate Modelling

4.4 Type Systems
   4.4.1 Primitive Types
   4.4.2 Derived Types
   4.4.3 Type Systems in the Model Driven Architecture

4.5 Domain Modelling
   4.5.1 ER Diagrams
   4.5.2 UML Class Diagrams
   4.5.3 XML Schema
   4.5.4 EMF Ecore

4.6 Design Patterns for Data Access
   4.6.1 Database Broker
   4.6.2 Iterator

4.7 Lifecycle Modelling
   4.7.1 Implementation-Level Examples

4.8 Users and Access Control
   4.8.1 Access Control Lists
   4.8.2 Discretionary Access Control
   4.8.3 Mandatory Access Control
   4.8.4 Role-based Access Control

4.9 Describing Reusable Patterns within a Metamodel
   4.9.1 Independence of the Reusable Pattern Metamodel
   4.9.2 Implementation of the Reusable Pattern Metamodel

4.10 Conclusion

5 The Internet Application Modelling Language

5.1 Applying Software Engineering Principles
   5.1.1 Requirements Planning
   5.1.2 Modelling Language Development Approach
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.3 Software Process Model</td>
<td>87</td>
</tr>
<tr>
<td>5.1.4 Evaluation</td>
<td>88</td>
</tr>
<tr>
<td>5.1.5 Design Goals</td>
<td>88</td>
</tr>
<tr>
<td>5.2 Metamodel Design Principles</td>
<td>89</td>
</tr>
<tr>
<td>5.2.1 Addressing Security Risks through Modelling Language Design</td>
<td>89</td>
</tr>
<tr>
<td>5.2.2 Language Design Principles</td>
<td>89</td>
</tr>
<tr>
<td>5.2.3 Hierarchical Modelling Approach</td>
<td>91</td>
</tr>
<tr>
<td>5.2.4 Guidelines for Metamodel Refactoring</td>
<td>92</td>
</tr>
<tr>
<td>5.3 Language Overview</td>
<td>93</td>
</tr>
<tr>
<td>5.3.1 Package Overview</td>
<td>95</td>
</tr>
<tr>
<td>5.4 Metamodel Core</td>
<td>96</td>
</tr>
<tr>
<td>5.4.1 Logic Model</td>
<td>97</td>
</tr>
<tr>
<td>5.4.2 Function Model</td>
<td>100</td>
</tr>
<tr>
<td>5.4.3 Event-Condition-Action Model</td>
<td>101</td>
</tr>
<tr>
<td>5.4.4 An example decomposition of a Simple Condition</td>
<td>103</td>
</tr>
<tr>
<td>5.4.5 Wires Model</td>
<td>105</td>
</tr>
<tr>
<td>5.4.6 Constructs Model</td>
<td>105</td>
</tr>
<tr>
<td>5.5 Type System</td>
<td>106</td>
</tr>
<tr>
<td>5.5.1 Primitive Types</td>
<td>107</td>
</tr>
<tr>
<td>5.5.2 Additional Built-in Primitive Types</td>
<td>109</td>
</tr>
<tr>
<td>5.5.3 Value Instances</td>
<td>110</td>
</tr>
<tr>
<td>5.5.4 Client-side Input Validation</td>
<td>111</td>
</tr>
<tr>
<td>5.6 Domain Modelling</td>
<td>112</td>
</tr>
<tr>
<td>5.6.1 Domain Types</td>
<td>112</td>
</tr>
<tr>
<td>5.6.2 Design Decisions on Modelling Domain Types</td>
<td>114</td>
</tr>
<tr>
<td>5.6.3 Domain Iterators</td>
<td>116</td>
</tr>
<tr>
<td>5.6.4 Domain Instances</td>
<td>119</td>
</tr>
<tr>
<td>5.6.5 Design Decisions on Modelling Domain Instances</td>
<td>120</td>
</tr>
<tr>
<td>5.7 Events</td>
<td>122</td>
</tr>
<tr>
<td>5.7.1 onChange</td>
<td>122</td>
</tr>
<tr>
<td>5.7.2 onInput</td>
<td>122</td>
</tr>
<tr>
<td>5.7.3 onAccess</td>
<td>123</td>
</tr>
<tr>
<td>5.7.4 onInit</td>
<td>123</td>
</tr>
<tr>
<td>5.7.5 onClick</td>
<td>123</td>
</tr>
<tr>
<td>5.7.6 onSent</td>
<td>123</td>
</tr>
<tr>
<td>5.7.7 onFailure</td>
<td>124</td>
</tr>
<tr>
<td>5.7.8 onIterate</td>
<td>124</td>
</tr>
<tr>
<td>5.7.9 Client-side Events</td>
<td>124</td>
</tr>
<tr>
<td>5.8 Operations</td>
<td>125</td>
</tr>
<tr>
<td>5.8.1 Modelling Complex Behaviour</td>
<td>125</td>
</tr>
<tr>
<td>5.8.2 Activity Operations</td>
<td>128</td>
</tr>
<tr>
<td>5.8.3 Activity Predicates</td>
<td>130</td>
</tr>
<tr>
<td>5.8.4 Replacing the Default Behaviour of Model Elements</td>
<td>131</td>
</tr>
<tr>
<td>5.9 Wires</td>
<td>131</td>
</tr>
</tbody>
</table>
Contents

5.9.1 Sync Wire .................................... 132
5.9.2 Set Wire ..................................... 133
5.9.3 Detail Wire ................................... 133
5.9.4 Autocomplete Wire ............................... 134

5.10 Users and Access Control ................................ 136
5.10.1 Roles and Permissions ............................. 136
5.10.2 Login Handlers ................................. 137
5.10.3 Access Control Handlers ............................ 138
5.10.4 OpenID ..................................... 139

5.11 Scopes .......................................... 140
5.11.1 Internet Application .............................. 140
5.11.2 Session ..................................... 141
5.11.3 Failure Handlers ................................ 141
5.11.4 Gates ...................................... 142
5.11.5 Query Parameter ................................ 143
5.11.6 Discussion ................................... 143

5.12 Messaging ........................................ 143
5.12.1 Email ...................................... 144
5.12.2 Additional Messaging Types ......................... 144

5.13 User Interface Modelling ................................ 145
5.13.1 Existing Approaches in Modelling User Interface Types ........... 145
5.13.2 Visible Thing .................................. 146
5.13.3 Sample Representations ............................ 150
5.13.4 Frame ...................................... 150

5.14 Internet Application ................................... 150

5.15 Visual Modelling Metaphors .............................. 151
5.15.1 Overview Layer ................................ 151
5.15.2 Navigation Layer ................................ 152
5.15.3 User Interface Layer .............................. 152
5.15.4 Domain Modelling Layer ............................ 152
5.15.5 Operation Modelling Layer ........................... 152

5.16 Unification of Type Instantiation and Classification .................. 153
5.16.1 Guidelines for Applying Instance Unification in Metamodels......... 155
5.16.2 Applying Instance Unification to the IAML Metamodel ............. 156

5.17 Conclusion ....................................... 157

6 Implementation Technologies ................................ 159
6.1 Common Comparison Criteria ................................ 159
6.1.1 Execution Environment Comparison Properties ...................... 159
6.1.2 Open Source Comparison Criteria ............................. 160

6.2 Metamodelling Environments ................................ 161
6.2.1 Eclipse Modeling Framework ................................ 162
6.2.2 ArgoUML ...................................... 163
6.2.3 Whitehorse ..................................... 164
6.2.4 Marama ....................................... 166
7.5.1 Completion ................................... 211
7.5.2 Rule Set ..................................... 211
7.5.3 Drools ...................................... 214
7.6 Code Generation .................................... 214
7.6.1 openArchitectureWare ..................... 214
7.6.2 Templates .................................... 215
7.6.3 Runtime Library ................................ 216
7.6.4 Platform Configuration ..................... 217
7.6.5 Output Formatter ................................ 217
7.6.6 Generator .................................... 218
7.7 Model Verification ................................. 219
7.7.1 Model Instance Verification with Checks .......... 220
7.7.2 Model Instance Verification with OCL in the EMF Validation Framework .......... 221
7.7.3 Model Instance Verification with CrocoPat .......... 221
7.7.4 Model Instance Verification with NuSMV .......... 222
7.8 Model Actions .................................... 222
7.9 Tests ........................................... 223
7.9.1 Model-driven Code Coverage ................... 224
7.10 Conclusion ........................................ 225

8 Evaluation ........................................... 227
8.1 Feature Comparison ................................. 227
8.2 Modelling Requirements ............................. 229
8.3 Benchmarking Application Implementation ............ 231
8.3.1 Implementation in Symfony .................... 232
8.3.2 Implementation in IAML ....................... 233
8.3.3 Re-implementation in Symfony ................. 252
8.3.4 System Metrics Evaluation ..................... 254
8.4 Metamodelling Metrics ............................. 255
8.5 Visual Model Evaluation ............................ 257
8.5.1 Notation Information Capacity .................. 258
8.5.2 Cognitive Dimensions Evaluation ............... 259
8.6 Conclusion ........................................ 259

9 Conclusions and Future Research .................. 261
9.1 Research Contributions and Conclusions ............. 261
9.1.1 Requirements for Modelling Rich Internet Applications ..................... 261
9.1.2 A Benchmarking Application for Rich Internet Applications .......... 261
9.1.3 A Modelling Language for Rich Internet Applications ............... 262
9.1.4 Model Completion .............................. 263
9.1.5 Model Instance Verification ..................... 264
9.1.6 Evaluation of Model-Driven Technologies .......... 264
9.1.7 Other Contributions .............................. 265
9.2 Answers to Research Questions ........................ 265
9.3 Future Research .................................. 267
9.3.1 Further Evaluation of Existing Web Modelling Languages ........ 267
9.3.2 Modelling Full RIAs ........................................ 268
9.3.3 Improved Graphical Editor View Mappings .................... 268
9.3.4 Integrating Textual Expression Languages in the Visual Editor .... 269
9.3.5 Extraction of Reusable Components ........................ 269
9.3.6 Code Generation Templates for Additional Platforms .......... 270
9.3.7 Incremental Transformations ................................ 270
9.3.8 Extensibility of the Proof-of-Concept Implementation of IAML .... 270
9.3.9 User Evaluations on the Proof-of-Concept Implementation of IAML 271
9.3.10 Identifying Metamodel Refactoring Patterns ................... 271
9.4 Summary .................................................. 273

A Use Cases .................................................. 275
A.1 Actors .................................................. 275
A.2 List of Use Cases ........................................ 276
  UC-01 View Data ........................................... 277
  UC-02 Update Data ......................................... 277
  UC-03 Pagination .......................................... 278
  UC-04 User Action Auditing ................................ 278
  UC-05 Debug Mode ......................................... 279
  UC-06 Server Transaction Support ........................... 279
  UC-07 Local Data Storage .................................. 280
  UC-08 Server Data Access .................................. 281
  UC-09 Persistent Client Data ................................. 281
  UC-10 Temporary Server Data ................................ 282
  UC-11 Uploading Files ...................................... 283
  UC-12 Restore Server Session ................................ 284
  UC-13 User Authorisation ................................... 285
  UC-14 Password Reset ...................................... 286
  UC-15 Session Support ...................................... 286
  UC-16 Account Registration .................................. 287
  UC-17 Automatic User Authorisation .......................... 288
  UC-18 Static Views (HTML) .................................. 289
  UC-19 Asynchronous Form Validation .......................... 290
  UC-20 Client Form Validation ................................ 291
  UC-21 Server Form Validation ................................ 292
  UC-22 Multiple Browser Support .............................. 293
  UC-23 Mobile Phone Support ................................ 293
  UC-24 Remote Data Source ................................... 294
  UC-25 Active Remote Data Source ............................. 294
  UC-26 Data Feeds ........................................... 295
  UC-27 Web Service .......................................... 295
  UC-28 Back/Forwards Button Control .......................... 296
  UC-29 Opening New Windows .................................. 296
  UC-30 Client-Side Application ................................ 297
| UC-31 | Communication with Software | 298 |
| UC-32 | Mobile Phone Communication | 299 |
| UC-33 | E-mailing Users | 300 |
| UC-34 | E-mail Unsubscription | 301 |
| UC-35 | Persistent Errors | 302 |
| UC-36 | User Content Access Control | 302 |
| UC-37 | Private User Content Access Control | 303 |
| UC-38 | User Collaboration | 304 |
| UC-39 | Interactive Map | 305 |
| UC-40 | Drag and Drop | 305 |
| UC-41 | Client Timer Support | 306 |
| UC-42 | Server Timer Support | 307 |
| UC-43 | Page Caching | 307 |
| UC-44 | Offline Application Support | 308 |
| UC-45 | Loading Time Support | 308 |
| UC-46 | Flash MP3 Support | 309 |
| UC-47 | Flash Communication Support | 310 |
| UC-48 | Internationalisation Support | 310 |
| UC-49 | Logout Control | 311 |
| UC-50 | Single Sign-In Solutions | 311 |
| UC-51 | User Redirection | 312 |
| UC-52 | Keyboard Shortcuts | 313 |
| UC-53 | Undo/Redo Support | 313 |
| UC-54 | Browser-Based Chat | 314 |
| UC-55 | Pop-up Window Support | 315 |
| UC-56 | Incompatible Client Warning | 315 |
| UC-57 | Dynamic Objects | 316 |
| UC-58 | Store Data in Local Database | 317 |
| UC-59 | Store Resources Locally | 318 |
| UC-60 | Multiple Client Threads | 319 |
| UC-61 | Multiple Server Threads | 320 |
| UC-62 | Communication with Plugins | 321 |
| UC-63 | Scheduled Events | 321 |
| UC-64 | Custom API Publishing | 322 |
| UC-65 | Runtime Interface Updates | 322 |
| UC-66 | Out-of-Order Events | 323 |
| UC-67 | Backwards-Compatible Scripting | 323 |
| UC-68 | Spellchecking | 324 |
| UC-69 | Autocomplete | 324 |

B Documentation By Example: Sync Wire 325

C OpenBRR Evaluations of Model-driven Technologies 331

D Model Checking using NuSMV: Implementing the Infinitely Redirects Constraint 335
E Description of the Attached Media 339
   E.1 Media Contents ........................................... 339
   E.2 IAML Tutorial .............................................. 340
       E.2.1 Creating a New Model Instance ...................... 340
       E.2.2 Editing the Model Instance .......................... 340
       E.2.3 Generating the Application .......................... 343
       E.2.4 Configure the Web Server ............................ 343
       E.2.5 Access the Generated Application .................... 343
F XMI Representation of Ticketiaml 347

G Language-specific Metrics of the Ticket 2.0 Implementations 357

H Cognitive Dimensions Evaluation of IAML 361

I Model Element Reference 367
   I.1 Access Control Handler .................................... 368
   I.2 Accessible .............................................. 369
   I.3 Action .................................................. 369
   I.4 Action Edge Source ...................................... 370
   I.5 Activity Node ........................................... 370
   I.6 Activity Operation ...................................... 371
   I.7 Activity Parameter ...................................... 373
   I.8 Activity Predicate ....................................... 374
   I.9 Arithmetic ............................................. 375
   I.10 Autocomplete Wire ....................................... 376
   I.11 Boolean Property ....................................... 377
   I.12 Builtin Operation ...................................... 377
   I.13 Builtin Property ........................................ 379
   I.14 Button ............................................... 380
   I.15 Can Be Synced .......................................... 381
   I.16 Cancel Node ............................................ 382
   I.17 Cast Node ............................................. 383
   I.18 Changeable ............................................. 383
   I.19 Complex Term .......................................... 384
   I.20 Condition Edge Destination ............................. 385
   I.21 Constraint Edge ........................................ 385
   I.22 Constraint Edge Destination ............................ 386
   I.23 Constraint Edges Source ................................ 386
   I.24 Contains Functions ...................................... 386
   I.25 Contains Operations .................................... 387
   I.26 Contains Values ........................................ 387
   I.27 Contains Wires ......................................... 388
   I.28 Data Flow Edge ......................................... 389
   I.29 Data Flow Edge Destination ............................ 389
   I.30 Data Flow Edges Source ................................ 390
I.31 Decision Node ..................................... 390
I.32 Detail Wire ....................................... 391
I.33 Domain Attribute .................................... 392
I.34 Domain Attribute Instance ............................... 392
I.35 Domain Feature ..................................... 393
I.36 Domain Feature Instance ............................... 394
I.37 Domain Instance .................................... 394
I.38 Domain Iterator ..................................... 395
I.39 Domain Source ..................................... 400
I.40 Domain Type ...................................... 401
I.41 ECA Rule ........................................ 402
I.42 EXSD Data Type .................................... 403
I.43 Email .......................................... 404
I.44 Event .......................................... 405
I.45 Execution Edge ..................................... 407
I.46 Execution Edge Destination .............................. 407
I.47 Execution Edges Source ................................ 408
I.48 Extends Edge ...................................... 408
I.49 Extends Edge Destination ................................ 409
I.50 Extends Edges Source .................................. 409
I.51 External Value...................................... 410
I.52 Finish Node ....................................... 410
I.53 Frame ........................................... 411
I.54 Function ......................................... 412
I.55 Gate ........................................... 413
I.56 Generated Element ................................... 414
I.57 Generates Elements ................................... 415
I.58 Input Form ....................................... 416
I.59 Input Text Field ..................................... 417
I.60 Internet Application ................................... 418
I.61 Iterator List ....................................... 419
I.62 Join Node ........................................ 419
I.63 Label .......................................... 420
I.64 Login Handler ...................................... 421
I.65 Map ........................................... 425
I.66 Map Point ........................................ 425
I.67 Message ......................................... 426
I.68 Named Element ..................................... 426
I.69 Operation ........................................ 427
I.70 Operation Call Node .................................. 427
I.71 Parameter ........................................ 427
I.72 Parameter Edge Destination .............................. 430
I.73 Parameter Edges Source ................................ 430
I.74 Parameter Value ..................................... 431
I.75 Permission ........................................ 431
I.76 Predicate ........................................ 431
I.77 Provides Edge ...................................... 432
I.78 Provides Edge Destination ............................... 432
I.79 Provides Edges Source ................................. 433
I.80 Query Parameter .................................... 433
I.81 Requires Edge ...................................... 433
I.82 Requires Edge Destination ............................... 434
I.83 Requires Edges Source ................................. 435
I.84 Role ........................................... 435
I.85 Schema Edge ...................................... 436
I.86 Scope .......................................... 436
I.87 Select Edge ....................................... 438
I.88 Session ......................................... 439
I.89 Set Node ......................................... 440
I.90 Set Wire ......................................... 440
I.91 Simple Condition .................................... 444
I.92 Split Node ........................................ 445
I.93 Start Node ........................................ 446
I.94 Sync Wire ........................................ 446
I.95 Temporary Variable ................................... 449
I.96 Value .......................................... 449
I.97 Visible Thing ...................................... 451
I.98 Wire ........................................... 452
I.99 Wire Destination ..................................... 452
I.100 Wire Source ....................................... 453
I.101 Wireable ......................................... 453
I.102 XQuery Function .................................... 454
I.103 XQuery Predicate .................................... 454

Bibliography ........................................ 455
## List of Tables

1.1 A brief history of Web application technologies ........................................ 2

2.1 Fundamental modelling requirements of Rich Internet Applications (1) ........ 18
2.2 Fundamental modelling requirements of Rich Internet Applications (2) .......... 19
2.3 Existing modelling language support for the general feature categories of modelling Rich Internet Applications ........................................ 25
2.4 Selected metamodelling metrics for metamodels within the Eclipse Modeling Framework ........................................ 28
2.5 Overall system metrics ........................................ 29
2.6 Language system metrics ........................................ 29

3.1 A list of popular metamodels and their meta-metamodels ........................... 40
3.2 The relationships between different modelling approaches .......................... 51
3.3 Information encoding capacity of different visual variables .......................... 60

4.1 Selected event algebra operators for composing complex events ................. 69
4.2 JUnit lifecycle methods and annotations ........................................ 79
4.3 Possible lifecycle layers of Rich Internet Applications and their potential use ...... 81
4.4 Example of an Access Matrix ........................................ 82

5.1 Additional Requirements for Full RIAs ........................................ 86
5.2 Associations between first-order logic elements and IAML elements ............ 98
5.3 A selection of permitted, conditional and prohibited datatype casts in IAML ... 109
5.4 Sample visual representations of interface modelling elements in IAML ....... 149
5.5 Visual Metaphors for a Rich Internet Application modelling language ........... 152

6.1 Requirement comparisons between existing metamodelling environments .... 167
6.2 Requirement comparisons between existing graphical modelling environments ... 172
6.3 Requirement comparisons between existing code generation environments .... 176
6.4 Requirement comparisons between existing model completion rule engines .... 181
6.5 Requirement comparisons between existing model verification environments (1) ... 192
6.6 Requirement comparisons between existing model verification environments (2) ... 193
6.7 A summary of the suitability of verification languages for addressing each verification approach category ........................................ 193

7.1 Generated IAML diagram editors and their associated OSGi bundle IDs ....... 201
7.2 Shape styles for the visual representation of IAML model elements ............... 203
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3</td>
<td>Background colours for the visual representation of IAML model elements</td>
<td>204</td>
</tr>
<tr>
<td>7.4</td>
<td>Summary of the rules used for model completion on IAML model instances</td>
<td>213</td>
</tr>
<tr>
<td>7.5</td>
<td>The set of platform-specific configuration properties used by the IAML Code Generator component</td>
<td>218</td>
</tr>
<tr>
<td>8.1</td>
<td>Reuse of existing metamodels in the IAML metamodel</td>
<td>228</td>
</tr>
<tr>
<td>8.2</td>
<td>A comparison of IAML against existing modelling language support for the general feature categories of modelling Rich Internet Applications</td>
<td>229</td>
</tr>
<tr>
<td>8.3</td>
<td>Evaluation of the modelling requirements of Basic RIAs against IAML (1)</td>
<td>230</td>
</tr>
<tr>
<td>8.4</td>
<td>Evaluation of the modelling requirements of Basic RIAs against IAML (2)</td>
<td>231</td>
</tr>
<tr>
<td>8.5</td>
<td>Legend to the modelling requirements evaluation of IAML</td>
<td>231</td>
</tr>
<tr>
<td>8.6</td>
<td>Comparing two Ticket 2.0 implementations using system metrics: overall development effort</td>
<td>254</td>
</tr>
<tr>
<td>8.7</td>
<td>Comparing two Ticket 2.0 implementations using system metrics: manual effort</td>
<td>254</td>
</tr>
<tr>
<td>8.8</td>
<td>Using metamodel metrics to evaluate the IAML metamodel against other similar metamodels implemented using the Eclipse Modeling Framework</td>
<td>256</td>
</tr>
<tr>
<td>8.9</td>
<td>Selected metamodeling metrics for metamodels within the Eclipse Modeling Framework</td>
<td>256</td>
</tr>
<tr>
<td>8.10</td>
<td>Evaluation of the information encoding capacity of IAML visual notation</td>
<td>259</td>
</tr>
<tr>
<td>C.1</td>
<td>Evaluations of model-driven technologies against OpenBRR ratings (1)</td>
<td>332</td>
</tr>
<tr>
<td>C.2</td>
<td>Evaluations of model-driven technologies against OpenBRR ratings (2)</td>
<td>333</td>
</tr>
<tr>
<td>C.3</td>
<td>Evaluations of model-driven technologies against OpenBRR ratings (3)</td>
<td>334</td>
</tr>
<tr>
<td>G.1</td>
<td>File-based metrics of the development of Ticketsf within Symfony 1.0.12</td>
<td>358</td>
</tr>
<tr>
<td>G.2</td>
<td>File-based metrics of the development of Ticketsf-mini within Symfony 1.4.8</td>
<td>359</td>
</tr>
<tr>
<td>G.3</td>
<td>File-based metrics of the development of Ticketiaml within IAML 0.6</td>
<td>360</td>
</tr>
</tbody>
</table>
List of Figures

1.1 The general methodology of design science research .......................... 5
2.1 A reasonable architecture for Rich Internet Applications ....................... 10
2.2 A sample WebML hypertext model ............................................. 21
2.3 A sample UWE navigation structure model ..................................... 22
2.4 The conceptual structure of the Ticket 2.0 benchmarking application .......... 27
3.1 The Metamodelling Architecture of MDA ....................................... 39
3.2 The Viewpoint Architecture of MDA ............................................ 41
3.3 The layers of UML under the MDA ............................................... 42
3.4 Using both architectures of the MDA to describe the model-driven development of RIAs 44
3.5 Dealing with different Modelling Spaces ....................................... 45
3.6 The Model Transformation Pattern in MDA .................................... 46
3.7 Using change models for round-trip engineering ................................ 47
3.8 The model completion process within model-driven development .............. 52
3.9 Definition of the rule program $C$ for the default checkbox rule convention .... 54
3.10 Model completion $C(A)$ ............................................................. 54
4.1 Managing complexity through horizontal partitioning ............................ 64
4.2 Managing complexity through hierarchical modelling ............................ 64
4.3 Managing complexity through aspect-oriented modelling ....................... 65
4.4 Event-Condition-Actions in UML activity diagrams ............................ 67
4.5 An AORML Interaction Pattern diagram ....................................... 67
4.6 Modelling Reaction Rules in URML ............................................. 68
4.7 Visual Event Handler definitions in Kaitiaki .................................... 68
4.8 Operation modelling using UML activity diagrams ............................ 70
4.9 Operation modelling of an online shopping business process using BPMN ..... 71
4.10 The built-in datatype hierarchy of XML Schema Datatypes .................... 73
4.11 Representing XML Schema Datatypes within the metamodelling architecture of MDA using modelling spaces .............................. 74
4.12 A sample ER Diagram and Data Object Table ................................ 75
4.13 A sample UML class diagram ..................................................... 76
4.14 The Database Broker design pattern represented as a UML sequence diagram 78
4.15 The Iterator design pattern using parameterised types ........................ 78
4.16 Component lifecycle events in OSGi ............................................ 81
5.1 The hybrid modelling language process model used in the development of IAML .............................................. 87
5.2 Hierarchical Modelling in IAML ......................................................................................................................... 91
5.3 An overview of some of the important concepts of modelling RIAs using IAML ......................................................... 94
5.4 The generated implementation of the IAML overview model instance ......................................................................... 94
5.5 Dependencies between packages of related model elements within IAML ........................................................................ 96
5.6 Layered architecture diagram illustrating the layered design of packages within the IAML metamodel ................................................................. 97
5.7 A partial syntax of first-order logic .......................................................................................................................... 98
5.8 UML class diagram for the Core Package: Logic Model .......................................................................................... 99
5.9 UML class diagram for the Core Package: Function Model .................................................................................. 100
5.10 UML class diagram for the Core Package: Event-Condition-Action Model .......................................................... 102
5.11 Ticketiaml: An ECA Rule connecting an Event to an Operation .......................................................................... 103
5.12 The Simple Condition decomposition of a single predicate using variables and domain types as terms, represented in terms of UML syntax .......................................................... 104
5.13 The model developer view of the Simple Condition decomposition ........................................................................ 104
5.14 VisualAge for Smalltalk: Using Connections ....................................................................................................... 105
5.15 UML class diagram for the Core Package: Wires Model .................................................................................. 105
5.16 UML class diagram for the Core Package: Constructs Model .................................................................................. 106
5.17 Adapting the metamodelling architecture of MDA to the use of primitive type systems ............................................. 107
5.18 UML class diagram for the Domain package of IAML .......................................................................................... 112
5.19 Ticketiaml: The defined Domain Attributes of the Domain Type Event ................................................................ 113
5.20 If domain types are defined using metamodelling, the resulting models are incompatible in terms of the MDA metamodelling architecture .................................................................................. 115
5.21 Defining instances using references instead of instantiation is MDA-compatible ......................................................... 116
5.22 UML class diagram for the Domain Instances package of IAML ........................................................................ 118
5.23 Ticketiaml: Selecting an instance of the Event Domain Type through a Domain Iterator connected to a Domain Source .................................................................................................................. 119
5.24 Ticketiaml: Creating a new instance of an Event using a Domain Iterator ................................................................ 121
5.25 Representing complex behaviour within IAML using two ECA Rules ........................................................................ 126
5.26 UML class diagram for the Activity Package: Operations Model ........................................................................ 128
5.27 The same complex behaviour of Figure 5.25 expressed with an Activity Operation .............................................. 129
5.28 UML class diagram for the Activity Package: Execution Model ........................................................................ 130
5.29 UML class diagram for the Activity Package: Data Flow Model ........................................................................ 131
5.30 UML class diagram for the Activity Package: Predicates Model ........................................................................ 132
5.31 UML class diagram for the Wires package of IAML .......................................................................................... 132
5.32 Ticketiaml: Connecting a Detail Wire to a Domain Iterator .................................................................................. 134
5.33 An instance of autocomplete in Gmail .................................................................................................................. 135
5.34 Autocomplete implemented in IAML using an Autocomplete Wire ......................................................................... 135
5.35 The user interface for an Autocomplete Wire, as generated by the proof-of-concept implementation of IAML .................................................................................................................. 136
5.36 UML class diagram for the Users package of IAML .......................................................................................... 137
5.37 UML class diagram for the Access Control Package .......................................................................................... 138
5.38 Ticketiaml: Protecting access to a Scope via an Login Handler and Access Control Handler .................................................. 139
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.39</td>
<td>UML class diagram for the Scopes package of IAML</td>
<td>141</td>
</tr>
<tr>
<td>5.40</td>
<td>Ticketiaml: Protecting access to a Scope via an entry Gate</td>
<td>142</td>
</tr>
<tr>
<td>5.41</td>
<td>UML class diagram for the Messaging package of IAML</td>
<td>144</td>
</tr>
<tr>
<td>5.42</td>
<td>Instantiation of user interface elements in WebML</td>
<td>145</td>
</tr>
<tr>
<td>5.43</td>
<td>Instantiation of user interface elements in UWE</td>
<td>146</td>
</tr>
<tr>
<td>5.44</td>
<td>UML class diagram for the Visual package of IAML</td>
<td>147</td>
</tr>
<tr>
<td>5.45</td>
<td>Ticketiaml: Designing the user interface for the Browse Events page using Visible Things</td>
<td>151</td>
</tr>
<tr>
<td>5.46</td>
<td>The consequences of unifying instance types within the metamodeling architecture of the MDA</td>
<td>153</td>
</tr>
<tr>
<td>5.47</td>
<td>A proposed refactoring of the IAML metamodel to unify the instantiation of Domain Instances and Visible Things</td>
<td>154</td>
</tr>
<tr>
<td>6.1</td>
<td>Using ArgoUML to define a UML class diagram</td>
<td>165</td>
</tr>
<tr>
<td>6.2</td>
<td>Defining a metamodel instance using Marama</td>
<td>166</td>
</tr>
<tr>
<td>6.3</td>
<td>The GMF Framework Wizard in Eclipse</td>
<td>170</td>
</tr>
<tr>
<td>6.4</td>
<td>The Shape Designer of Marama showing the shape design and corresponding concrete view</td>
<td>171</td>
</tr>
<tr>
<td>7.1</td>
<td>Overall UML component diagram of the Proof-of-concept Implementation of IAML</td>
<td>196</td>
</tr>
<tr>
<td>7.2</td>
<td>UML component diagram of the IAML Model Component Decomposition</td>
<td>197</td>
</tr>
<tr>
<td>7.3</td>
<td>UML component diagram of the Graphical Editor Component Decomposition</td>
<td>199</td>
</tr>
<tr>
<td>7.4</td>
<td>Implementation of a graphical editor for IAML model instances using the Graphical Modeling Framework</td>
<td>200</td>
</tr>
<tr>
<td>7.5</td>
<td>Illustrating generated and overridden elements in the IAML editor</td>
<td>208</td>
</tr>
<tr>
<td>7.6</td>
<td>UML component diagram of the Model Completion Component Decomposition</td>
<td>210</td>
</tr>
<tr>
<td>7.7</td>
<td>UML component diagram of the Code Generation Component Decomposition</td>
<td>215</td>
</tr>
<tr>
<td>7.8</td>
<td>UML component diagram of the Model Verification Component Decomposition</td>
<td>219</td>
</tr>
<tr>
<td>7.9</td>
<td>Visualisation of model instance constraint violations within a diagram editor</td>
<td>220</td>
</tr>
<tr>
<td>7.10</td>
<td>Illustrating code coverage of code generation templates</td>
<td>225</td>
</tr>
<tr>
<td>8.1</td>
<td>A screenshot of the Browse Events page implemented in Ticketiaml</td>
<td>251</td>
</tr>
<tr>
<td>8.2</td>
<td>A screenshot of the View Event page implemented in Ticketiaml</td>
<td>251</td>
</tr>
<tr>
<td>8.3</td>
<td>A screenshot of the Browse Events page implemented in Ticketsf-mini</td>
<td>253</td>
</tr>
<tr>
<td>8.4</td>
<td>A screenshot of the View Event page implemented in Ticketsf-mini</td>
<td>253</td>
</tr>
<tr>
<td>8.5</td>
<td>Comparison of the number of classes (NoC) and abstract classes (NoAC) of the IAML metamodel against other similar metamodels</td>
<td>257</td>
</tr>
<tr>
<td>8.6</td>
<td>Comparison of the total number of references (TNoR) and attributes (TNoA) of the IAML metamodel against other similar metamodels</td>
<td>258</td>
</tr>
<tr>
<td>9.1</td>
<td>Two approaches in defining a Complex Term that references a Value contained within a Changeable as a Parameter</td>
<td>269</td>
</tr>
<tr>
<td>B.1</td>
<td>Ticketiaml: The completed contents of the edit event Domain Iterator</td>
<td>327</td>
</tr>
<tr>
<td>B.2</td>
<td>Ticketiaml: The completed contents of the Event-typed Domain Instance</td>
<td>327</td>
</tr>
<tr>
<td>B.3</td>
<td>Ticketiaml: The completed contents of the Edit Event Input Form</td>
<td>328</td>
</tr>
</tbody>
</table>
List of Figures

B.4 Ticketiaml: The completed contents of the title Input Text Field, within the Edit Event Input Form, to implement synchronisation ........................................... 328
B.5 A screenshot of the Edit Event page implemented in Ticketiaml ........................................... 329

E.1 Creating a new IAML model instance through the Eclipse wizard menu (1) ........... 341
E.2 Creating a new IAML model instance through the Eclipse wizard menu (2) ........... 341
E.3 The IAML visual editor for a new IAML model instance ........................................... 342
E.4 The default contents of the “Home” Frame within a new IAML model instance .......... 342
E.5 The tool palette of the IAML visual editor ................................................................. 342
E.6 The IAML visual editor representing a Frame containing two Input Text Fields connected by a Sync Wire ................................................................. 344
E.7 Generating the source code for an IAML model instance (1) ................................. 344
E.8 Generating the source code for an IAML model instance (2) ................................. 344
E.9 Screenshot of the generated tutorial application accessed through Mozilla Firefox .... 345
<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Part of a web application implemented in WebDSL</td>
</tr>
<tr>
<td>2</td>
<td>Implementation of the <em>acyclical class inheritance</em> constraint of UML class diagrams in OCL</td>
</tr>
<tr>
<td>3</td>
<td>The IDL definition of the MouseEvent interface</td>
</tr>
<tr>
<td>4</td>
<td>EAnnotation documentation in Ecore</td>
</tr>
<tr>
<td>5</td>
<td>Internet Application code generation using a JET template</td>
</tr>
<tr>
<td>6</td>
<td>Internet Application code generation using an OAW Xpand Template</td>
</tr>
<tr>
<td>7</td>
<td>Example rule implementation in Jena</td>
</tr>
<tr>
<td>8</td>
<td>Example rule implementation in Jess</td>
</tr>
<tr>
<td>9</td>
<td>Example rule implementation in Drools</td>
</tr>
<tr>
<td>10</td>
<td>Implementation of <em>infinitely redirects</em> in Drools</td>
</tr>
<tr>
<td>11</td>
<td>Implementation of <em>infinitely redirects</em> in Jena</td>
</tr>
<tr>
<td>12</td>
<td>Implementation of <em>infinitely redirects</em> in OWL 2 Full</td>
</tr>
<tr>
<td>13</td>
<td>Implementation of <em>infinitely redirects</em> in CrocoPat</td>
</tr>
<tr>
<td>14</td>
<td>Implementation of <em>infinitely redirects</em> in Alloy</td>
</tr>
<tr>
<td>15</td>
<td>Implementation of <em>infinitely redirects</em> in openArchitectureWare: Xtend</td>
</tr>
<tr>
<td>16</td>
<td>Implementation of <em>infinitely redirects</em> in openArchitectureWare: Checks</td>
</tr>
<tr>
<td>17</td>
<td>Implementation of <em>infinitely redirects</em> in OCL</td>
</tr>
<tr>
<td>18</td>
<td>Implementation of <em>infinitely redirects</em> in LTL</td>
</tr>
<tr>
<td>19</td>
<td>One Drools rule used in the model completion implementation of the <em>Sync Wire</em> model element</td>
</tr>
<tr>
<td>20</td>
<td>Implementation of the <em>runFrameEvents</em> code generation template in Xpand</td>
</tr>
<tr>
<td>21</td>
<td>Implementation of the <em>containingScope</em> model extension in Xtend</td>
</tr>
<tr>
<td>22</td>
<td>Implementation of a Checks constraint for <em>Sync Wires</em></td>
</tr>
<tr>
<td>23</td>
<td>Implementation of an OCL constraint for <em>Values</em> in the EMF Verification Framework</td>
</tr>
<tr>
<td>24</td>
<td>Configuration details to host generated applications through the Apache HTTP Server</td>
</tr>
</tbody>
</table>